Data Structures and Algorithms in C++

(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

John Wiley & Sons

Solution of Exercise R-2.1

There are two immediate inefficiencies: (1) the chaining of constructors implies a potentially long set of function calls any time an instance of a deep class, Z, is created, and (2) the algorithm for determining which version of a certain function to use could end up looking through a large number of classes before it finds the right one to use.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

John Wiley & Sons

Solution of Exercise R-2.2

Whenever a large number of classes all extend from a single class, it is likely that you are missing out on potential code reuse from similar functions in different classes. There is likely some factoring of functions into common classes that could be done in this case, which would save programmer time and maintenance time, by eliminating duplicated code.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.3

Air traffic control software, computer integrated surgery applications, and flight navigation systems.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

John Wiley & Sons

Solution of Exercise R-2.4

Computer game development, online services for downloading music and movies, and telephone and other communication services over the internet.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.5

Many text editors dedicated for "tools". Member functions could include bringing up programs to perform spell checking, grammar checking, setting various editor preferences (such as the language for a word processor or tabbing and indenting options for text editors for programming languages), importing (and exporting) data from various sources, and tracking document changes.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.7

A class may have many different constructors. The compiler cannot know which of the base class's constructors would be appropriate to be called, or what arguments should be passed to it. So the derived class must make the call explicitly. There is only one destructor for any class, and so there is no choice involved. Further, since destructors are not explicitly called by the user (they are invoked automatically by the system), there is no way for a derived class to even invoke it base class's destructor.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.8

Because the class FibonacciProgression does not provide public access to the firstValue and nextValue member functions, we design a new class, FibonacciPlus, which extends the Fibonacci progression class by providing a public member function that returns the kth element of the sequence. It is assumed that $k \geq 1$.

```
class FibonacciPlus : public FibonacciProgression {
public:
                                        // constructor
  FibonacciPlus(long f = 0, long s = 1)
  : FibonacciProgression(f, s) { }

  long getKthValue(int k) {             // get the kth value (assumed k >= 1)
    long f = firstValue();              // get the first value
    if (k == 1) return f;               // return first if k == 1
    for (int i = 2; i < k; i++)         // get values 2 through (k-1)st values
      nextValue();
    return nextValue();                 // return kth value
  }
};

// ...
FibonacciPlus F(3, 4);                  // Fibonacci sequence from 3, 4
cout << F.getKthValue(7) << endl;       // output 7th element
```

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.9

2^{56} calls to nextValue will end on the value 2^{63}. Since the maximum positive value of a long is $2^{63} - 1$, $2^{56} - 1$ calls to nextValue can be made before a long-integer overflow.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.10

The function to be called is GeomProgression::firstValue(), and hence the value 1 is returned. The reason is that the function firstValue is declared to be virtual (in Progression), which means that the actual type of the object, and not the declared type of the pointer, is used to determine which member function is to be called.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.11

No, d is referring to a Equestrian object that is not also of type Racer. Casting in an inheritance relationship can only move up or down the hierarchy, not "sideways."

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.12

The two subclasses UndergraduateStudent and GraduateStudent are derived from the class Student, and thus inherit all the student members. In addition, the UndergraduateStudent class adds the new boolean member *inDorm*, which indicates whether the student lives in a dormitory. The GraduateStudent class adds the new string member *advisor*, which stores the name of the student's advisor.

The Faculty class is derived from Person, and the subclasses Professor and Instructor are derived from Faculty. The Faculty class adds new string members *officeNumber* and *phoneNumber*. The Instructor has a boolean variable *isPartTime*, which indicates whether the instructor is hire part-time. The Professor class adds a new boolean variable *isTenured*, which indicates

whether the professor is tenured.

```
class UndergraduateStudent : public Student {
private:
  bool inDorm;
  // ...
};
class GraduateStudent : public Student {
private:
  string advisor;
  // ...
};
class Faculty : public Person {
private:
  string officeNumber;
  string phoneNumber;
  // ...
};
class Professor : public Faculty {
private:
  bool isTenured;
  // ...
};
class Instructor : public Faculty {
private:
  bool isPartTime;
  // ...
};
```

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.13

Let us assume that we define an exception class ArrayIndexBounds, whose constructor is given the index value that caused the error. It has a member function badIndex that returns this index. The following code checks for the array subscript out of bounds, and if so, outputs an appropriate error message. (It is more likely that there is a class storing the array, and this class would throw the exception.)

```
try {
  if (i >= array.size()) throw ArrayIndexBounds(i);
  cout << array[i];
}
catch(ArrayIndexBounds& e) {
  cout << "Array index " << e.badIndex() << " is out of bounds.";
}
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise R-2.14

Read it.
Ship it.
Buy it.
Read it.
Box it.
Read it.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise C-2.1

The following program works by storing most of its source in a string variable, a. The program prints the initial (include) statements, then prints the contents of string a, but adding all the escape keys needed in the definition of a. Finally it prints the contents of a itself, which prints the remainder of the program. We define special characters for backslash (bs), double quotes (qq), and newline (nl), for help in handling the character escape sequences. There are shorter solutions than this, but they are generally harder to understand.

```
#include <iostream>
#include <string>
using namespace std;
string a=
"const char bs = '\\\\'; const char nl = '\\n'; const char qq = '\"';\n"
"main() {\n"
"  cout << \"#include <iostream>\" << endl;\n"
"  cout << \"#include <string>\" << endl;\n"
"  cout << \"using namespace std;\" << endl;\n"
"  cout << \"string a=\" << endl << qq;\n"
"  for (int i=0; i < a.size(); i++)\n"
"    switch(a[i]) {\n"
"      case nl: cout << bs << 'n' << qq << endl << qq; break;\n"
"      case bs: cout << bs << bs; break;\n"
"      case qq: cout << bs << qq; break;\n"
"      default: cout << a[i];\n"
"    }\n"
"  cout << qq << ';' << endl; cout << a << endl;\n"
"}";
const char bs = '\\'; const char nl = '\n'; const char qq = '"';
main() {
  cout << "#include <iostream>" << endl;
  cout << "#include <string>" << endl;
  cout << "using namespace std;" << endl;
  cout << "string a=" << endl << qq;
  for (int i=0; i < a.size(); i++)
    switch(a[i]) {
      case nl: cout << bs << 'n' << qq << endl << qq; break;
      case bs: cout << bs << bs; break;
      case qq: cout << bs << qq; break;
      default: cout << a[i];
    }
  cout << qq << ';' << endl; cout << a << endl;
}
```

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise C-2.4

Assume that lines are represented by the equation $y = ax + b$. The x coordinate of the intersection of this line with another line $y = a'x + b'$ is found by equating the right hand sides and solving for x, which yields $x = (b' - b)/(a - a')$. If $a = a'$ the lines are parallel, in which case we throw the Parallel exception.

```
class Parallel {                        // parallel intersection exception
private:
  string errMsg;
public:
  Parallel(const string& msg) { errMsg = msg; }
  string getMessage() const { return errMsg; }
};

class Line {
private:
  double a, b;                          // line is y = ax + b
public:
  Line(double aa = 0, double bb = 0) // constructor
    { a = aa; b = bb; }
  double intersect(const Line& ell) const throw(Parallel)
  {
    if (a == ell.a) throw Parallel("Intersection of parallel lines");
    return (ell.b - b)/(a - ell.a);
  }
};

int main() {
  Line L[3];
  L[0] = Line(2, 7);                    // create a few lines
  L[1] = Line(-3, -3);
  L[2] = Line(2, -11);
  for (int i = 0; i < 3; i++) {         // intersect all distinct pairs
    for (int j = 0; j < i; j++) {
      try {
        double x = L[i].intersect(L[j]);
        cout << "Lines " << j << " and " << i
             << " cross at " << x << endl;
      }
      catch (Parallel& e) {
        cout << e.getMessage() << endl;
      }
    }
  }
}
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount
John Wiley & Sons

Solution of Exercise C-2.5

```
class AbsValProgression : public Progression {
protected:
   long second;                          // second value of progression
   long prev;                            // previous value of progression

   long firstValue() {                   // resets to the first value
      cur = first;
      prev = first + second;
      return cur;
   }
   long nextValue() {                    // advances to the next value
      long temp = cur;
      if (prev > cur) cur = prev - cur;
      else            cur = cur - prev;
      prev = temp;
      return cur;
   }
public:
                                         // constructor
   AbsValProgression(long val1 = 2, long val2 = 200) {
      first = val1;
      second = val2;
   }
};
```

Data Structures and Algorithms in C++

(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

John Wiley & Sons

Solution of Exercise C-2.6

Even though the constructor takes an argument of type `double`, we cast the value to type `long`, since this is what the base class requires. In order to make the `sqrt` function accessible, we should add the statement "`#include <cmath>`".